

Chapter 10

Graphics & Animation

Labs

GOALS:

1. Students to create programs that involve animation and graphics.
2. Students to create programs that use audio.

ESSENTIAL QUESTIONS

1. Why are Graphics necessary for Visual Basic?
2. Why is it necessary to use animation in Visual Basic?

LESSON

Graphics is a very important part of visual basic programming because an attractive interface will be appealing to the users. There are four basic controls in VB6 that you can use to draw graphics on your form: the line control, the shape control, the image box and the picture box

The line and Shape controls

To draw a straight line, just click on the line control and then use your mouse to draw the line on the form. After drawing the line, you can then change its color, width and style using the **BorderColor**, **BorderWidth** and **BorderStyle** properties. Similarly, to draw a shape, just click on the shape control and draw the shape on the form. The default shape is a rectangle, with the default shape property set at 0. You can change the shape to square, oval, circle and rounded rectangle by changing the shape property's value to 1, 2, 3, 4, and 5 respectively. In addition, you can change its background color using the **BackColor** property, its border style using the **BorderStyle** property, its border color using the **BorderColor** property as well its border width using the **BorderWidth** property.

ForeColor – Changes the color of the text displayed on an object. Go to the properties list and select **ForeColor** then select **Palette** tab.

You can change the **BackColor** and **ForeColor** at runtime by using the assignment statement and the VB color constants **vbBlack**, **vbBlue**, **vbCyan**, **vbGreen**, **vbMagenta**, **vbRed**, **vbWhite**, **vbYellow**

Example:

`Me.BackColor = RGB(255,0,0) → RGB(Red, Green, Blue)` numbers are from 0 to 255 → Changes to Red

```
Private Sub Form_Load()  
    Me.BackColor = vbBlue  
End Sub
```

Image objects do not have color properties

Project 1

1. Create the following program
2. Answer the questions at the end of Project

' Change Background Color

Option Explicit

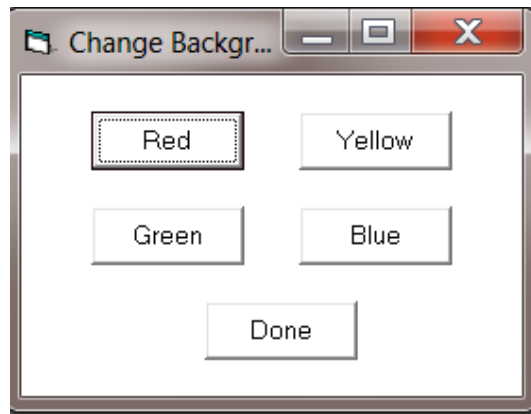
```
Private Sub cmdRed_Click()  
    Me.BackColor = vbRed  
End Sub
```

```
Private Sub cmdYellow_Click()  
    Me.BackColor = vbYellow  
End Sub
```

```
Private Sub cmdGreen_Click()  
    Me.BackColor = vbGreen  
End Sub
```

```
Private Sub cmdBlue_Click()  
    Me.BackColor = vbBlue  
End Sub
```

```
Private Sub cmdDone_Click()  
    Unload Me  
End Sub
```



Questions:

1. Can you use the form name instead of Me for the BackColor?
2. List all possible vbColors:

Adding Lines to an Application

A line object is created using the Line control in the Tool Box.

Properties of Line object:

Name – prefix lin

BorderColor – change color of line

BorderStyle – can be set to transparent, solid, dash, dot, dash-dot, dash-dot-dot, or inside-solid. Can be change with assignment using vbBSDash etc...

BorderWidth – Change the thickness of a line.

When creating lines, the BackColor of the form should be set first because changing the BackColor of the form erases any graphics drawn on the form.

Adding Shapes to a Program

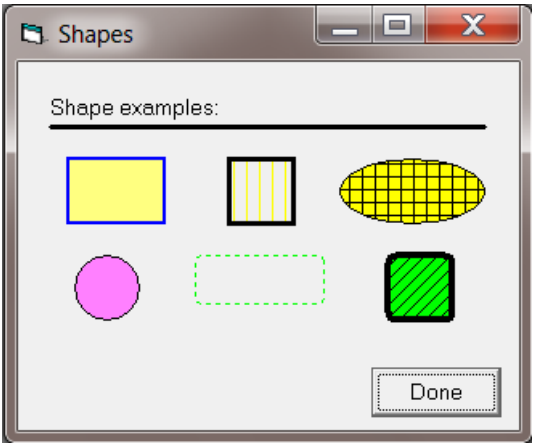
A shape can be added using the Shape control in the Tool box.

Shape properties


- Name – prefix shp
- BackColor – Changes the background color of a shape
- BackStyle – Transparent or opaque
- BorderColor – Changes the color of the outline of a shape
- BorderStyle – Can be set to the same style as a line
- BorderWidth – Changes the thickness of the outline of a shape
- FillColor – Changes the thickness of the outline of a shape
- FillStyle – Can be set to solid, transparent, horizontal line, vertical line, upward diagonal, downward diagonal, cross, or diagonal cross.
- Shape – can be set to rectangle, square, oval, circle, rounded rectangle, or rounded square. Can be change by the assignment operator example vbShapeOval.

Project 2

1. Create the following program
2. Rectangle Blue Border Yellow Fill
3. Square Dark Black Border Light Yellow fill
4. Oval – Yellow Fill
5. Circle – Pink Fill
6. Dashed Green Rectangle
7. Square with diagonal lines and green fill
8. Print and turn in



Picture Box

Picture Boxes allow the user to place images and graphics on a form. 

Properties of a Picture Box:

- Name – prefix pic
- Picture – is used to display a dialog box for selecting image to display.
- BorderStyle – can be set to none (0) or fixed single (1)
- AutoSize – Can be true or false. True – picture box is resized to the same size of graphic. False graphic is resized to the same size as the picture box.
- Visible – True (Visible) or False (Hide)

The Image Box and the Picture Box

Using the line and shape controls to draw graphics will only enable you to create a simple design. In order to improve the look of the interface, you need to put in images and pictures of your own. Fortunately, there are two very powerful graphics tools you can use in Visual Basic 6 which are the image box and the picture box.

To load a picture or image into an image box or a picture box, you can click on the picture property in the properties window to launch a dialog box that will prompt you to select a certain picture file. You can also load a picture at runtime by using the LoadPicturure () method. The syntax

The code for Figure 2

```
Dim a, b, c As Integer
Private Sub Command1_Click ()
```

is

```
Image1.Picture= LoadPicture("C:\path name\picture file  
name") or
```

```
picture1.Picture= LoadPicture("C:\path name\picture  
name")
```

For example, the following statement will load the
grape.gif picture into the image box.

```
Image1.Picture= LoadPicture("C:\My Folder\VB  
program\Images\grape.gif")
```

Example 2

In this example, each time you click on the 'change
pictures' button as shown in Figure 2, you will be able to
see three images loaded into the image boxes. This
program uses the Rnd function to generate random integers
and then uses the LoadPicture method to load different
pictures into the image boxes using the
If...Then...Statements based on the random numbers
generated.

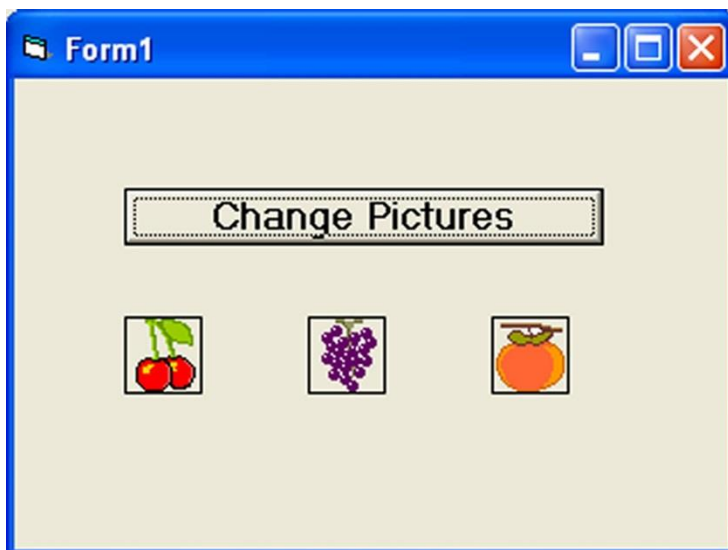


Figure 2

Randomize Timer

```
a = 3 + Int(Rnd * 3)  
b = 3 + Int(Rnd * 3)  
c = 3 + Int(Rnd * 3)
```

If a = 3 Then

```
Image1(0).Picture = LoadPicture("C:\My  
Folder\VB program\Images\grape.gif")
```

End If

If a = 4 Then

```
Image1(0).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\cherry.gif")
```

End If

If a = 5 Then

```
Image1(0).Picture = LoadPicture("C:\My  
Folder\VB program\Images\orange.gif")
```

End If

If b = 3 Then

```
Image1(1).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\grape.gif")
```

End If

If b = 4 Then

```
Image1(1).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\cherry.gif")
```

End If

If b = 5 Then

```
Image1(1).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\orange.gif")
```

End If

If c = 3 Then

```
Image1(2).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\grape.gif")
```

End If

If c = 4 Then

```
Image1(2).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\cherry.gif")
```

End If

If c = 5 Then

```
Image1(2).Picture = LoadPicture("C:\My  
Folder\VB  
program\Images\orange.gif")
```

End If

End Sub

Moving Shapes and Picture Boxes

A Shape or graphic can be moved within its container, which is a form or a picture box.

Properties of Shapes and Picture Boxes:

- Left – changes the position of the left edge of the object and the left edge of the form
- Top - changes the position of the top edge of the object and the top edge of the form
- Width – Changes the width of the object
- Height – Changes the height of the object.

Twips – The distance an object moves. 1,440 twips equal 1 inch

Move Method

The move method is used at run time to move a picture box or shape object to a new location.

Example: Object.Move left,top,width,height

Where left is a numeric value that specifies the twips between left edge of object and the left edge of the container. Top, width, & height are optional

Example: picSmiley.Move picSmiley.Left – 20

Moves the picture box named picSmiley 20 twips to the left of its current position.

Project 3

1. Create the following program.
2. Answer the Questions at the end.

' Move Graphic

Option Explicit

```
Private Sub cmdUp_Click() ' moves object 40 twips up
    picSmiley.Move picSmiley.Left, picSmiley.Top - 40
End Sub
```

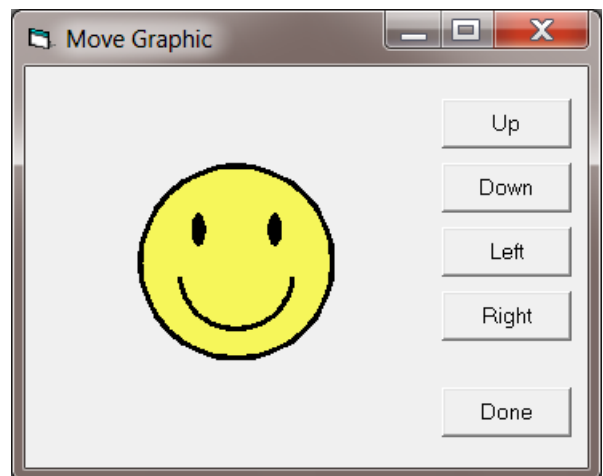
```
Private Sub cmdDown_Click() ' moves object 40 twips
down from its current position
```

```
    picSmiley.Move picSmiley.Left, picSmiley.Top + 40
End Sub
```

```
Private Sub cmdLeft_Click() ' moves object 40 twips to the left of its current position
    picSmiley.Move picSmiley.Left - 40
End Sub
```

```
Private Sub cmdRight_Click() ' moves object 40 twips to the right of its current position
    picSmiley.Move picSmiley.Left + 40
End Sub
```

```
Private Sub cmdDone_Click()
    Unload Me
End Sub
```



Questions:

1. What does twips stand for?
2. What is the difference between -40 and +40 twips?

Project 4

1. Create the following program.
2. Answer the following questions.

' Move Graphics

Option Explicit

Dim intTop As Integer, intRight As Integer,
intBottom As Integer, intLeft As Integer

Private Sub Form_Load()

' Change the drive and directory so that this
' application can find the data file(s) it uses:

ChDrive "C" ' Change X to the working drive

ChDir "C:\PathToObjects" ' Change X to the working drive

' Change PATH to the working path

picBox(0).Picture = LoadPicture("Ivy.wmf")

picBox(2).Picture = LoadPicture("Guitar.wmf")

picBox(3).Picture = LoadPicture("Panther.wmf")

intTop = 0

intRight = 1

intBottom = 2

intLeft = 3

End Sub

Private Sub cmdClockwise_Click()

Dim intTemp As Integer

'move top graphic right & down

picBox(intTop).Move picBox(intTop).Left + 1000, picBox(intTop).Top + 1000

'move right graphic left & down

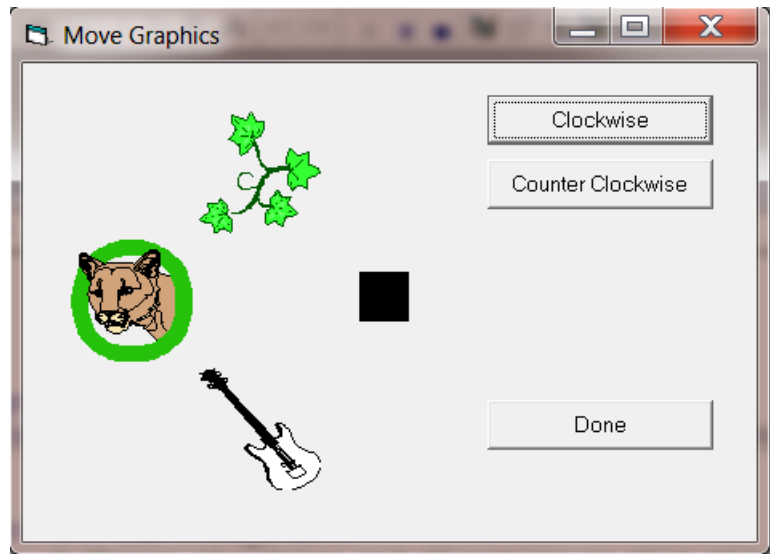
picBox(intRight).Move picBox(intRight).Left - 1000, picBox(intRight).Top + 1000

'move bottom graphic left & up

picBox(intBottom).Move picBox(intBottom).Left - 1000, picBox(intBottom).Top - 1000

'move left graphic right & up

picBox(intLeft).Move picBox(intLeft).Left + 1000, picBox(intLeft).Top - 1000



```

intTemp = intTop
intTop = intLeft
intLeft = intBottom
intBottom = intRight
intRight = intTemp
End Sub

Private Sub cmdCounterClockwise_Click()
    Dim intTemp As Integer

    'move top graphic left & down
    picBox(intTop).Move picBox(intTop).Left - 1000, picBox(intTop).Top + 1000
    'move right graphic left & up
    picBox(intRight).Move picBox(intRight).Left - 1000, picBox(intRight).Top - 1000
    'move bottom graphic right & up
    picBox(intBottom).Move picBox(intBottom).Left + 1000, picBox(intBottom).Top - 1000
    'move left graphic right & down
    picBox(intLeft).Move picBox(intLeft).Left + 1000, picBox(intLeft).Top + 1000

    intTemp = intBottom
    intBottom = intLeft
    intLeft = intTop
    intTop = intRight
    intRight = intTemp
End Sub
Private Sub cmdDone_Click()
    Unload Me
End Sub

```

Questions:

1. What is the purpose of ChDrive?
2. What is the purpose of ChDir?

Using the Graphic Methods

Other than using the line and shape controls to draw graphics on the form, you can also use the Pset, Line and Circle methods to draw graphics on the form.

(a) The Pset Method

The Pset method draw a dot on the screen, it takes the format

Pset (x , y), color

(x,y) is the coordinates of the point and color is its color. To specify the color, you can use the color codes or the standard VB color constant such as VbRed, VbBlue, VbGreen and etc. For example, Pset(100,200), VbRed will display a red dot at the (100,200) coordinates.

The Pset method can also be used to draw a straight line on the form. The procedure is

For x= a to b

Pset(x,x)

Next x

This procedure will draw a line starting from the point (a,a) and to the point (b,b). For example, the following procedure will draw a magenta line from the point (0,0) to the point (1000,1000).

For x= 0 to 100

Pset(x,x) , vbMagenta

Next x

(b) The Line Method

Although the Pset method can be used to draw a straight line on the form, it is a little slow. It is better to use the Line method if you want to draw a straight line faster. The format of the Line command is shown below. It draws a line from the point (x1, y1) to the point (x2, y2) and the color constant will determine the color of the line.

Line (x1, y1)-(x2, y2), color

For example, the following command will draw a red line from the point (0, 0) to the point (1000, 2000).

Line (0, 0)-(1000, 2000), VbRed

The Line method can also be used to draw a rectangle. The format is

Line (x1-y1)-(x2, y2), color, B

The four corners of the rectangle are (x1-y1), (x2-y1), (x1-y2) and (x2, y2)

Another variation of the Line method is to fill the rectangle with a certain color. The format is

Line (x1, y1)-(x2, y2), color, BF

If you wish to draw the graphics in a picture box, you can use the following formats

- Picture1.Line (x1, y1)-(x2, y2), color
- Picture1.Line (x1-y1)-(x2, y2), color, B
- Picture1.Line (x1-y1)-(x2, y2), color, BF
- Picture1.Circle (x1, y1), radius, color

(c) The Circle Method

Example: object.Circle(x,y), radius, color, start, end, aspect ratio where the arguments after radius are optional

The circle method takes the following format

Object.Circle (x1, y1), radius, color or Me.Circle(x1,y1)

That draws a circle centered at (x1, y1), with a certain radius and a certain border color. For example, the procedure

```
Circle (400, 400), 500, VbRed
```

draws a circle centered at (400, 400) with a radius of 500 twips and a red border.

Animation is always an interesting and exciting part of programming. Although visual basic is not designed to handle advance animations, you can still create some interesting animated effects if you put in some hard thinking. There are many ways to create animated effects in VB6, but for a start we will focus on some easy methods.

The simplest way to create animation is to set the **VISIBLE** property of a group of images or pictures or texts and labels to true or false by triggering a set of events such as clicking a button. Let's examine the following example:

This is a program that create the illusion of moving the jet plane in four directions, North, South ,East, West. In order to do this, insert five images of the same picture into the form. Set the visible property of the image in the center to be true while the rest set to false. On start-up, a user will only be able to see the image in the center. Next, insert four command buttons into the form and change the labels to Move North, Move East, Move West and Move South respectively. Double click on the move north button and key in the following procedure:

```
Sub Command1_click( )
```

```
Image1.Visible = False
```

```
Image3.Visible = True
```

```
Image2.Visible = False
```

```
Image4.Visible = False
```

```
Image5.Visible = False
```

```
End Sub
```

By clicking on the move north button, only image 3 is displayed. This will give an illusion that the jet plane has moved north. Key in similar procedures by double clicking other command buttons. You can also insert an addition command button and label it as Reset and key in the following codes:

```
Image1.Visible = True
```

```
Image3.Visible = False
```

```
Image2.Visible = False
```

```
Image4.Visible = False
```

```
Image5.Visible = False
```

Clicking on the reset button will make the image in the center visible again while other images become invisible, this will give the false impression that the jet plane has move back to the original position.



You can also issue the commands using a textbox, the code is shown below:

```
Private Sub Command1_Click()  
    If Text1.Text = "n" Then  
        Image1.Visible = False  
        Image3.Visible = True  
        Image2.Visible = False  
        Image4.Visible = False  
        Image5.Visible = False  
    ElseIf Text1.Text = "e" Then  
        Image1.Visible = False  
        Image4.Visible = True  
        Image2.Visible = False  
        Image3.Visible = False  
        Image5.Visible = False  
    ElseIf Text1.Text = "w" Then  
        Image1.Visible = False  
        Image3.Visible = False  
        Image2.Visible = False  
        Image4.Visible = False  
        Image5.Visible = True  
    ElseIf Text1.Text = "s" Then  
        Image1.Visible = False  
        Image3.Visible = False  
        Image2.Visible = True  
        Image4.Visible = False  
        Image5.Visible = False  
    End If
```

End Sub

Another simple way to simulate animation in VB6 is by using the Left and Top properties of an object. Image.Left give the distance of the image in twips from the left border of the screen, and Image.Top give the distance of the image in twips from the top border of the screen, where 1 twip is equivalent to 1/1440 inch.

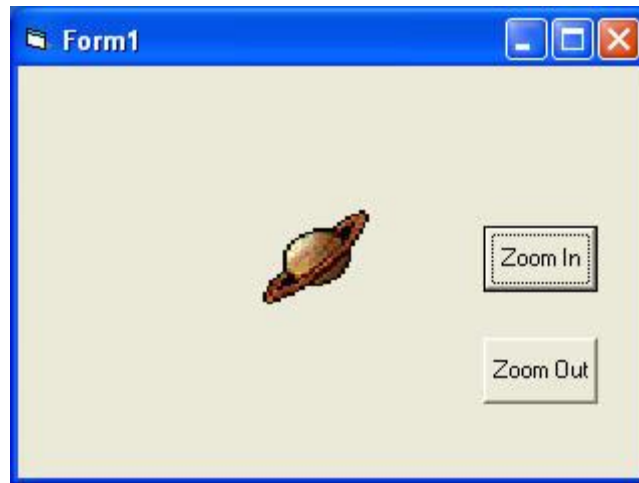
Using a statement such as `Image.Left-100` will move the image 100 twips to the left, `Image.Left+100` will move the image 100 twip away from the left(or 100 twips to the right), `Image.Top-100` will move the image 100 twips to the top and `Image.Top+100` will move the image 100 twips away from the top border (or 100 twips down).Below is a program that can move an object up, down. left, and right every time you click on a relevant command button.



The Code

```
Private Sub Command1_Click()  
    Image1.Top = Image1.Top + 100  
End Sub  
Private Sub Command2_Click()  
    Image1.Top = Image1.Top - 100  
End Sub  
Private Sub Command3_Click()  
    Image1.Left = Image1.Left + 100  
End Sub  
Private Sub Command4_Click()  
    Image1.Left = Image1.Left - 100  
End Sub
```

The fourth example let user magnify and diminish an object by changing the height and width properties of an object. It is quite similar to the previous example. The statements `Image1.Height = Image1.Height + 100` and `Image1.Width = Image1.Width + 100` will increase the height and the width of an object by 100 twips each time a user click on the relevant command button. On the other hand, The statements `Image1.Height = Image1.Height - 100` and `Image1.Width = Image1.Width -100` will decrease the height and the width of an object by 100 twips each time a user click on the relevant command button



The Code

```
Private Sub Command1_Click()  
    Image1.Height = Image1.Height + 100  
    Image1.Width = Image1.Width + 100  
End Sub
```

```
Private Sub Command2_Click()  
    Image1.Height = Image1.Height - 100  
    Image1.Width = Image1.Width - 100  
End Sub
```

You can try to combine both programs above and make an object move and increases or decreases in size each time a user click a command button.

PaintPicture Method

The PaintPicture Method form: `object.PaintPicture object2.Picture,x,y`
where object is the container and object2 is the name of an object that has a Picture property, and x & y are variables, values, or expressions that specify the upper-left position of picture.

Example:

```
Private Sub cmdPaintPicture_Click()  
    picNewLocation.PaintPicture imgSmiley.Picture, 100, 100  
End Sub
```

Cls Method

Used in a statement that clears the object in the container.

Example:

```
Object.Cls
```

CONTAINERS PROPERTIES THAT AFFECT GRAPHICS

Containers object have properties that affect the appearance of graphics drawn using graphic methods:

Method Properties:

AutoRedraw – True appears in containers. False it may not appear

DrawStyle – Can be vbSolid,vbDash,vbDot,vbDashDot, vbDashDotDot, vbInvisible, vbInsideSolid
 DrawWidth – Changes the thickness of the graphic's outline. Can use assignment statement 1 to 32,767
 FillColor – Changes inside color of a shape. Can use assignment statement to change color using color constant.
 FillStyle – can be vbFSSolid, vbFSTransparent, vbHorizontalLine, vbVerticalLine, vbUpwardDiagonal, vbDownwardDiagonal, vbCross, vbDiagonalCross.
 ForeColor – Changes the color of the graphic's outline.

Project 5

1. Create the following program
2. Answer questions at the end of project

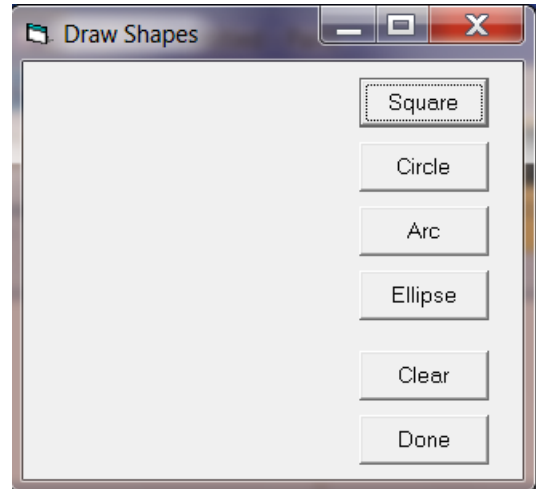
' Draw Shapes

Option Explicit

```

Private Sub cmdSquare_Click()
    Me.Line (100, 100)-(500, 500), , B
End Sub
Private Sub cmdCircle_Click()
    Me.Circle (1000, 800), 200
End Sub
Private Sub cmdArc_Click()
    Const Pi As Double = 3.14
    Me.Circle (1700, 1400), 200, , 0, Pi
End Sub
Private Sub cmdEllipse_Click()
    Me.Circle (2400, 1900), 400, , , 1.5
End Sub
Private Sub cmdClear_Click()
    Me.Cls
End Sub


Private Sub cmdDone_Click()
    Unload Me
End Sub
  
```



Questions:

1. Why did the circle and ellipse have commas after it with no values?
Example:Me.Circle (2400, 1900), 400, , , 1.5
2. Can the form name be used instead of Me?

USING A TIMER OBJECT

A timer object is used to execute code at specified intervals and also be used to simulate animation. Timer Object icon  in the Tool Box.

Timer Object Properties:

Name – prefix starts with tmr.

Interval – Specified in milliseconds. 1,000 milliseconds equal 1 second. Range between 0 to 64,767

Enabled – True → enable and False → disabled

A timer event occurs automatically after the amount of time specified in the Interval property has passes.

Moving Line Objects

Line objects created using the Line control can be moved at run time using the properties:

- X1 – Changes the x coordinate starting point, can be changed at run time through assignment.
- Y1 - Changes the y coordinate starting point, can be changed at run time through assignment.
- X2 - Changes the x coordinate end point, can be changed at run time through assignment.
- Y2 - Changes the y coordinate end point, can be changed at run time through assignment.

Project 6

1. Create the following program
2. Answer the questions at the end

' Move Line

Option Explicit

```
Private Sub Form_Load()  
    Randomize  
End Sub
```

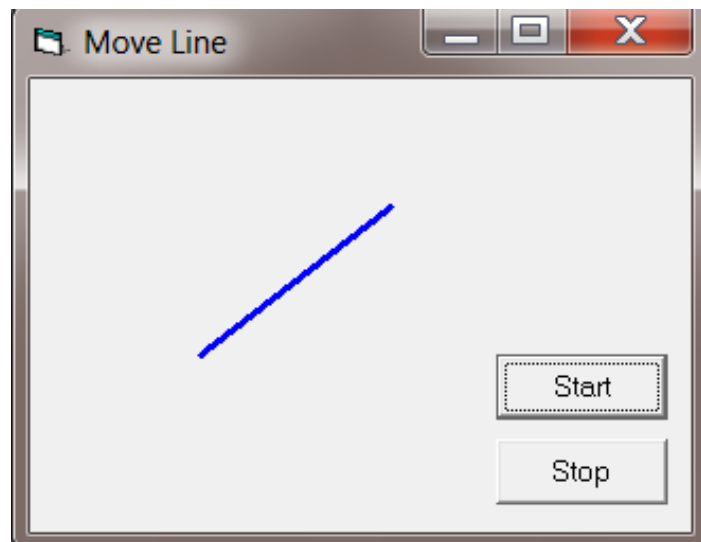
```
Private Sub cmdStart_Click()  
    tmrMoveLine.Enabled = True  
End Sub
```

```
Private Sub tmrMoveLine_Timer()  
    linMoveLine.X1 = Int(Me.Width * Rnd) 'high number is width of form  
    linMoveLine.Y1 = Int(Me.Height * Rnd) 'high number is height of form  
    linMoveLine.X2 = Int(Me.Width * Rnd)  
    linMoveLine.Y2 = Int(Me.Height * Rnd)  
End Sub
```

```
Private Sub cmdStop_Click()  
    Unload Me  
End Sub
```

Questions:

1. What affect did Rnd have on the line itself?




2. Why did they use Me.Height & Me.Width?

Animation using Timer Object

In order to make it move automatically, you need to use a timer. The first step in creating automatic animation is to drag the timer from the toolbox into the form and set its interval to a certain value other than 0. A value of 1 is 1 milliseconds which means a value of 1000 represents 1 second. The value of the timer interval will determine the speed on an animation.

In the following example, I use a very simple technique to show animation by using the properties Visible=False and Visible=true to show and hide two images alternately. When you click on the program, you should see the following animation.

A screenshot of a Windows application window titled 'Form1'. The window has a blue title bar with standard minimize, maximize, and close buttons. The main area is a light beige color. In the center of the form, there is a small, pixelated image of a dog, possibly a beagle, facing forward.

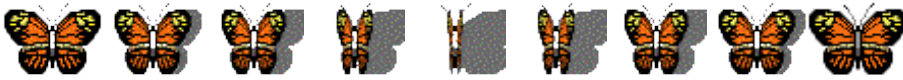
The Code

```
Private Sub Timer1_Timer()  
    If Image1.Visible = True Then  
        Image1.Visible = False  
        Image2.Visible = True  
    ElseIf Image2.Visible = True Then  
        Image2.Visible = False  
        Image1.Visible = True  
    End If  
End Sub
```

Next example shows a complete cycle of a motion such as the butterfly flapping its wing. Previous examples show only manual animation while this example will display an automatic animation once you start the program or by clicking a command button. You'll need to insert a group of eight images of a butterfly flapping its wings at different stages. Next, insert a timer into the form and set the interval to 10 or any value you like. Remember to make image1 visible while other images invisible at start-up. Finally, insert a command button; rename its caption as Animate and key in the following statements by double clicking on this button. Bear in mind that you should enter the statements for hiding and showing the images under the timer1_timer subroutine otherwise the animation would work. Clicking on the animate button make timer start ticking and the event will run after every interval of 10 milliseconds or whatever interval you have set at design time. In future lesson, I will show you how to adjust the interval at runtime by using a slider bar or a scroll bar. When you run the program, you should see the following animation:

Animation for a complete motion

So far those examples of animation shown in lesson 23 only involve movement of static images. In this lesson, you will be able to create true animation where an action finishes in a complete cycle, for example, a butterfly flapping its wings. In the following example, I used eight picture frames of a butterfly which display a butterfly flapping its wing at different stages.



You can actually copy the above images and use them in your program. You need to put all the above images overlapping one another, make image1 visible while all other images invisible at start-up. Next, insert a command button and label it as Animate. Click on the command button and key in the statements that make the images appear and disappear successively by using the properties image.visible=true and image.visible=false. I use If..... Then and ElseIf to control the program flow. When you run the program, you should be able to get the following animation.

The Interface

The Code

```
Private Sub Command1_Click()
```

```

If Image1.Visible = True Then
Image1.Visible = False
    Image2.Visible = True
ElseIf Image2.Visible = True Then
Image2.Visible = False
    Image3.Visible = True
ElseIf Image3.Visible = True Then
    Image3.Visible = False
    Image4.Visible = True
ElseIf Image4.Visible = True Then
    Image4.Visible = False
    Image5.Visible = True
ElseIf Image5.Visible = True Then
    Image5.Visible = False
    Image6.Visible = True
ElseIf Image6.Visible = True Then
    Image6.Visible = False
    Image7.Visible = True
ElseIf Image7.Visible = True Then
    Image7.Visible = False
    Image8.Visible = True
ElseIf Image8.Visible = True Then
    Image8.Visible = False
    Image1.Visible = True
End If

```

```
End Sub
```



If you wish to create the effect of the butterfly flapping its wing and flying at the same time, then you could use the Left and Top properties of an object. Below is an example of a subroutine where the butterfly will flap its

wing and move up at the same time. You can also write subroutines that move the butterfly to the left, to the right and to the bottom.

Sub move_up()

```
If Image1.Visible = True Then
    Image1.Visible = False
    Image2.Visible = True
    Image2.Top = Image2.Top - 100
Elseif Image2.Visible = True Then
    Image2.Visible = False
    Image3.Visible = True
    Image3.Top = Image3.Top - 100
Elseif Image3.Visible = True Then
    Image3.Visible = False
    Image4.Visible = True
    Image4.Top = Image4.Top - 100
Elseif Image4.Visible = True Then
    Image4.Visible = False
    Image5.Visible = True
    Image5.Top = Image5.Top - 100
Elseif Image5.Visible = True Then
    Image5.Visible = False
    Image6.Visible = True
    Image6.Top = Image6.Top - 100
Elseif Image6.Visible = True Then
    Image6.Visible = False
    Image7.Visible = True
    Image7.Top = Image7.Top - 100
Elseif Image7.Visible = True Then
    Image7.Visible = False
    Image8.Visible = True
    Image8.Top = Image8.Top - 100
Elseif Image8.Visible = True Then
    Image8.Visible = False
    Image1.Visible = True
    Image1.Top = Image1.Top - 100
End If
```

End Sub

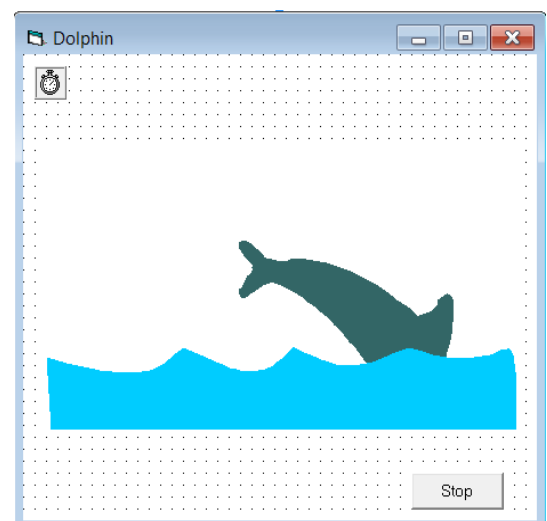
Project 7

1. Create the following program
2. Used the images provided
3. Answer the following questions

' Dolphin

Option Explicit

Private intIncrement As Integer



```
Private Sub Form_Load()  
    picDolphin(0).Visible = True ' displays first picture box  
    picDolphin(1).Visible = False ' hides second picture box  
    picDolphin(2).Visible = False ' hides third picture box  
    intIncrement = 0  
End Sub
```

```
Private Sub tmrDolphinJump_Timer()  
    picDolphin(intIncrement).Visible = False ' hides currently displayed picture box  
    If intIncrement = 2 Then ' determines next picture box to display  
        intIncrement = 0  
    Else  
        intIncrement = intIncrement + 1  
    End If  
    picDolphin(intIncrement).Visible = True ' displays next picture box in sequence  
End Sub
```

```
Private Sub cmdStop_Click()  
    Unload Me  
End Sub
```

Questions:

1. What is the purpose of using the “if statements”?
2. What was the purpose of the following line of code? **picDolphin(intIncrement).Visible = True**